

## Contents

<b>1 Routine/Function Prologues</b>	<b>2</b>
1.0.1 setnoahp.F90 (Source File: setnoahp.F90) . . . . .	2

## 1 Routine/Function Prologues

### 1.0.1 setnoahp.F90 (Source File: setnoahp.F90)

This subroutine retrieves NOAH parameters - Significant F90 revisions below this subroutine will be required in the future.

#### REVISION HISTORY:

28 Apr 2002: Kristi Arsenault; Added NOAH LSM, Initial Code  
 13 Oct 2003: Sujay Kumar; Domain independent modifications

#### INTERFACE:

```
subroutine setnoahp
```

#### USES:

```
use lisdrv_module, only : grid,tile,lis
use noah_varder      ! NOAH tile variables
#ifndef ( defined OPENDAP )
  use opendap_module
#endif
```

#### CONTENTS:

```
#if ( ! defined OPENDAP )
  integer :: tnroffset = 0
#endif
!-----
! Convert UMD Classes to SIB Classes for Each Tile
!-----
print*, 'MSG: setnoahp -- Calling MAPVEGC to convert UMD to SIB', &
      (' , iam, ')
print*, 'DBG: setnoahp -- nch',lis%d%ncnch,' ( ,iam, )'

do n=1,lis%d%ncnch
  call mapvegc(tile(n)%vegt)
  noah(n)%vegt = tile(n)%vegt
enddo
print*, 'DBG: setnoahp -- left MAPVEGC', (' ,iam, ')
!-----
! Get Vegetation Parameters for NOAH Model in Tile Space
! Read in the NOAH Static Vegetation Parameter Files
!-----
open(unit=11,file=noahdrv%noah_vfile,status='old')

do j=1,noahdrv%noah_nvegp
  read(11,*)(value(i,j),i=1,lis%p%nt)
enddo
close(11)
```

```

!-----
! Assign STATIC vegetation parameters to each tile based on the
! type of vegetation present in that tile.
! These parameters will be stored in one long array--structured
! as follows: Tile 1, all the parameters (1 through numparam)
! then Tile 2, all the parameters
! Then Tile 3, all the parameters etc.
!-----

do i=1,lis%d%nch
  do j=1,noahdrv%noah_nvegp
    noah(i)%vegp(j)=value(tile(i)%vegt,j)
  enddo
enddo
!-----
! Read in bottom temperature fields and adjust for elevation differnce
! with either Eta (NLDAS) or GDAS (GLDAS) correction datasets.
!-----
#if ( defined OPENDAP )
  allocate(placetbot(parm_nc,1+nroffset:parm_nr+nroffset))
#else
  allocate(placetbot(lis%d%gnc,lis%d%gnr))
#endif

#if ( defined OPENDAP )
  print*, 'MSG: setnoahp -- Retrieving TBOT file ', &
    trim(noahdrv%noah_tbot), ' (,iam,)'
  call system("opendap_scripts/gettbot.pl "//ciam// " //&
    trim(noahdrv%noah_tbot)//" //&
    cparm_slat//" //cparm_nlat//" //&
    cparm_wlon//" //cparm_elon)
#endif
print *, 'Opening GLDAS TBOT File ', noahdrv%NOAH_TBOT
OPEN(UNIT=12, FILE=noahdrv%NOAH_TBOT, &
  FORM="UNFORMATTED")

READ(12) PLACETBOT
CLOSE(12)

print*, 'MSG: setnoahp -- Read TBOT file', (',iam,)'
do i = 1, lis%d%nch
  if(placetbot(tile(i)%col,tile(i)%row-tnroffset).ne.-9999.00) then
    noah(i)%tempbot = placetbot(tile(i)%col, tile(i)%row-tnroffset)
  endif
enddo
deallocate(placetbot)
call absoft_release_cache()

```

```

!-----
! The MAX SNOW ALBEDO field is opened and read in here:
!-----

#if ( defined OPENDAP )
  allocate(tmpalb(parm_nc,1+nroffset:parm_nr+nroffset))
#else
  allocate(tmpalb(lis%d%gnc,lis%d%gnr))
#endif

#if ( defined OPENDAP )
  print*, 'MSG: setnoahp -- Retrieving MAXSNALB file ', &
    trim(noahdrv%noah_mxsnal), '(,iam,)'
  call system("opendap_scripts/getmaxsnalb.pl "//ciam//" //&
    trim(noahdrv%noah_mxsnal)//" //&
    cparm_slat//" //cparm_nlat//" //&
    cparm_wlon//" //cparm_elon)
#endif

print *, 'MSG: setnoahp -- Opening GLDAS MAXSNALB File', &
  trim(noahdrv%noah_mxsnal), '(,iam,)'
OPEN(UNIT=12,FILE=noahdrv%NOAH_MXSNAL, &
  FORM="UNFORMATTED")

READ(12) TMPALB
CLOSE(12)

do i=1,lis%d%nch
  if(tmpalb(tile(i)%col, tile(i)%row-tnroffset).ne.-9999.00) then
    noah(i)%mxsnalb = tmpalb(tile(i)%col, tile(i)%row-tnroffset)
  endif
enddo

deallocate(tmpalb)
call absoft_release_cache()

print*, 'MSG: setnoahp -- Read mxsnalb', '(,iam,)'
print*, 'MSG: setnoahp -- reading soil and clay files', '(,iam,)'
!-----
! Open soil files (sand, clay, and porosity).
!-----

#if ( defined OPENDAP )
  allocate(sand1(parm_nc,parm_nr))
  allocate(clay1(parm_nc,parm_nr))
  print*, 'MSG: setnoahp -- Retrieving SAND file ', &
    trim(lis%p%safile), '(,iam,)'
  call system("opendap_scripts/getsand.pl "//ciam//" //&
    trim(lis%p%safile)//" //&
    cparm_slat//" //cparm_nlat//" //&
    cparm_elon)
#endif

```

```

        cparm_wlon//" "//cparm_elon)
print*, 'MSG: setnoahp -- Retrieving CLAY file ', &
       trim(lis%p%clfile), '(,iam,)'
call system("opendap_scripts/getclay.pl "//ciam//" //&
           trim(lis%p%clfile)//" //"
                     &
           cparm_slat//" "//cparm_nlat//" //"
                     &
           cparm_wlon//" "//cparm_elon)
#endif
else
  allocate(sand1(lis%d%gnc,lis%d%gnr))
  allocate(clay1(lis%d%gnc,lis%d%gnr))
endif
OPEN(15,FILE=LIS%P%SAFILE,FORM='UNFORMATTED',STATUS='OLD')
OPEN(16,FILE=LIS%P%CLFILE,FORM='UNFORMATTED',STATUS='OLD')
!-----
!   Read soil properties and convert to Zobler soil classes.
!   Note that the 3 files each contain 3 global records, one
!   for each layer depth (0-2, 2-150, 150-350 cm). At this
!   time only the top layer data are used to map soil
!   parameters. Since NOAH has 4 layers, the third layer of
!   porosity is temporarily used for layer 4 as well.
!-----
READ(15) SAND1
READ(16) CLAY1
CLOSE(15)
CLOSE(16)
print*, 'MSG: setnoahp -- read sand and clay files', '(,iam,)'
!-----
! Determine Zobler-equivalent soil classes derived from
! percentages of sand and clay, used in NOAH.
!-----
#if ( defined OPENDAP )
  allocate(soiltyp(parm_nc,1+nroffset:parm_nr+nroffset))
  CALL SOILTYPE(parm_nc,parm_nr,SAND1,CLAY1,SOILTYP)
#else
  allocate(soiltyp(lis%d%gnc,lis%d%gnr))
  CALL SOILTYPE(LIS%D%GNC,LIS%D%GNR,SAND1,CLAY1,SOILTYP)
#endif

  deallocate(sand1)
  deallocate(clay1)
  call absoft_release_cache()
!-----
! Read in the NOAH Soil Parameter File
!-----
OPEN(UNIT=18,FILE=noahdrv%NOAH_SFILE,STATUS='OLD', &
      ACCESS='SEQUENTIAL')

DO I=1,noahdrv%NOAH_NSOILP

```

```

      READ(18,*)(BASICSET(JJ,I),JJ=1,noahdrv%NOAH_ZST)
      ENDDO
      CLOSE(18)
      print*, 'MSG: setnoahp -- read sfile ', trim(noahdrv%NOAH_SFILE), &
            ', iam, )'
!-----
! Convert grid space to tile space for soil type values.
!-----
      allocate(placesltyp(lis%d%nch))
      do i=1,lis%d%nch
         placesltyp(i) = soiltyp(tile(i)%col, tile(i)%row-tnroffset)
         noah(i)%zobsoil = placesltyp(i)
      end do

      if(lis%o%wparam.eq.1) then
         allocate(stype(lis%d%lnc,lis%d%lnr))
         do i=1,lis%d%nch
            if(grid(i)%lat*1000.ge.lis%d%kgds(4).and. &
               grid(i)%lat*1000.le.lis%d%kgds(7).and. &
               grid(i)%lon*1000.ge.lis%d%kgds(5).and. &
               grid(i)%lon*1000.le.lis%d%kgds(8)) then
               rindex = tile(i)%row - (lis%d%kgds(4)-lis%d%kgds(44)) &
                  /lis%d%kgds(9)
               cindex = tile(i)%col - (lis%d%kgds(5)-lis%d%kgds(45)) &
                  /lis%d%kgds(10)
               stype(cindex,rindex) = noah(i)%zobsoil(1)*1.0
            endif
         enddo

         open(32,file="soiltype.bin",form='unformatted')
         write(32) stype
         close(32)
         deallocate(stype)
      endif

      deallocate(soiltyp)
      call absoft_release_cache()
!-----
! Assign SOIL Parameters to each tile based on the
! type of Zobler soil class present in that tile.
!-----
      DO I=1,lis%d%nch          !Tile loop
         K=PLACESLTYPI(I)        !Soil type
         DO J=1,noahdrv%NOAH_NSOILP   !Soil parameter loop
            NOAH(I)%SOILP(J)=BASICSET(K,J)
         ENDDO !J
      ENDDO !I
      deallocate(placesltyp)

```

```
call absoft_release_cache()
return
```